Worst case bin packing for OTN electrical layer networks dimensioning

Tamás Király*, Attila Bernáth*, László Végh*, Lajos Bajzik, Erika Kovács*, Kristóf Bérczi*, Alpár Jüttner*, Tibor Jordán***

* Dept. of Operations Research, Eötvös University, Budapest
e-mail: {tkiraly,athos,veghal,koverika,berkri,alpar,jordan}@cs.elte.hu,
and
** Nokia Siemens Networks
Budapest, Hungary
e-mail: lajos.bajzik@nsn.com

Abstract

The *OTN (Optical Transport Network)* standard, defined by ITU-T Recommendation G.709 and G.872, contains a flexible digital hierarchy of *ODU (Optical Data Unit)* signals. The ODU hierarchy provides sub-wavelength grooming in OTN networks, which is necessary for efficient utilization of the high bit rates of optical channels. When dimensioning the links of a transport network consisting of ODU switches, the packing of lower order ODU signals into higher order ODU signals needs to be taken into account. These networks are expected to be controlled by *GMPLS (Generalized MPLS)*, which puts specific constraints on the dimensioning. We assume that there is no explicit label control and that the *GMPLS control plane is using first-fit strategy for making reservations on a link*. With these assumptions the link dimensioning problem is defined as deciding how many higher order ODU component links are required on an OTN GMPLS bundled link for first-fit packing of a given set of lower order ODU demands, in any order of arrival. The paper provides strict bounds for ODU hierarchy-specific item and bin sizes. Then, it introduces an extended variant of the dimensioning problem, when lower order ODU connections which are not controlled by GMPLS are also present.

Keywords: OTN, maximum resource packing, GMPLS

1 Introduction

The OTN standard is described in ITU-T Recommendations G.709 and G.872 [1, 2]. An OTN network can have an electrical grooming layer on top of the optical domain, which provides transport of *lower order ODU (LO-ODU)* demands. The electrical grooming layer consists of ODU switches connected by links which are provided by optical channels. These links are bundles of component links. One component link is an *OTU (Optical Transport Unit)* connection. Each OTU is mapped into one optical channel. An OTU component link of type OTUk either wraps a *higher order ODU (HO-ODU)* of type ODUk which carries a multiplex of LO-ODU demands, or wraps the signal of a single LO-ODU demand of type ODUk. The higher order and lower order ODU are relative terms, a lower order ODUj signal is a client of a higher order ODUk signal, $j < k$. We are considering only networks with single-stage ODU multiplexing. In this case the switching nodes multiplex or map the LO-ODU demands directly into the HO-ODU/OTU component links, without intermediate ODU stages.

This paper contributes packing results for the following ODU types: ODU0, ODU1, ODU2, ODU2e, ODU3 and ODU4. Beside these, the G.709 recommendation also defines the variable-sized ODUflex type. It is not covered in the paper, because ODUflex can have any discrete size, thus only general packing approximations can be used in this case, like the one described in [6] for *SDH (Synchronous Digital Hierarchy)*. There are two other types, ODU3e1 and ODU3e2 which are defined in [3], but not standardized. They are not covered either.

We consider only homogeneous links which contain component links of the same ODUk/OTUk type. In the rest of the paper we refer to this as the *bin type of the link*. The payload capacity of an ODUk consists of a standardized number of *Tributary Slots (TS)*, which will be referred to as *bin size*. The bin sizes of the covered bin types are shown in second column of Table 1. A LO-ODUj occupies a specific number of TS inside a HO-ODUk. We refer to this as the *item size* of the LO-ODUj demand with respect to the specific HO-ODUk. The right part of Table 1 contains the item sizes for the covered demand types. For the same demand type as the bin type we set the item size equal to the bin size, modeling that an OTUk component link can be used for carrying a single ODUk demand.

The basic link dimensioning problem is calculating the minimum number of component links which are required to accommodate a given set of demands routed on the link. A demand cannot be split between component links, thus we need to solve a bin packing problem. Furthermore, it is expected that ODU layer networks are controlled by GMPLS control plane [4]. In this case the demands are provisioned by distributed LSP (Label Switched Path)

Table 1: Bin and item sizes for the covered cases

Bin type	Bin size	Item sizes per demand type					
		ODU0	ODU1	ODU2	ODU2e	ODU3	ODU4
ODU1	2	1	2	–	–	–	–
ODU2	8	1	2	8	–	–	–
ODU3	32	1	2	8	9	32	–
ODU4	80	1	2	8	8	31	80

setup procedures. The GMPLS standard allows two options for finding the component link and tributary slots for a demand on a link. In the first option, it is decided locally by the GMPLS entity which controls the link. The second option is explicit label control, where the head-end of the LSP specifies it for each link along the LSP [5]. In this paper we cover the case when it is decided locally, i.e. there is no explicit label control. The local entity can follow several strategies for finding the component link to use for a new demand. We assume first-fit strategy. In this strategy the component links have a predefined order, and the first component link having enough free space to carry the demand is selected. Due to the distributed nature of GMPLS control plane, even if the set of demands to be carried by a link are known in advance, it is not predictable in which order they will be set up on the link. Thus the dimensioning needs to find the minimum number of component links which is enough for first fit-packing of the given demands in any possible order of arrival.

With the above assumptions, the link dimensioning problem is identical to the maximum resource bin packing problem [7]. An extension of the basic problem is when we allow demands that are not controlled by the GMPLS control plane beside demands controlled by GMPLS. We refer to these demands as non-GMPLS demands. The non-GMPLS demands are not packed with first-fit algorithm. They can be packed arbitrarily into the component links before the first-fit packing of GMPLS demands.

The rest of the paper structured as follows. Section 2 examines the situation when there are only GMPLS demands in the network (i.e. when all the demand are placed using the First Fit scheme). These results are then extended to the case of the existence of non-GMPLS demands in Section 3. Finally, the paper is concluded by mentioning some possible further extensions to the problems.

2 GMPLS demands

In mathematical terms, the problem is the following. Items of different sizes arrive in a certain order and are allocated to bins of the same size $b \in \mathbb{N}$ using a First Fit algorithm (that is, the arriving item is inserted in the first bin with enough space). We know the sizes of the items to allocate but we do not know their order. The problem is to find an order of the items that uses the most number of bins (i.e. a *worst-case* order), or just to tell the maximum number of bins needed in the worst case. This is already a difficult problem [9]: it is already NP-compete to decide whether the maximum number of bins is 2 or 3. The best known result compared to the best packing is that the worst-order first fit packing has an approximation factor of 12/7 (or a multiplicative factor of 17/10 and an additive term of 7/10) compared to the optimal packing [14]. Fortunately, in our cases the possible item-sizes are rather restricted, therefore we can give a much better upper bound for this maximum. But first let us see some general claims about the worst case orders.

2.1 Upper bounds for arbitrary item and bin sizes

Let $b \in \mathbb{N}$ be the bin size. The set of possible item-sizes is $S \subset \mathbb{N}$, and for every size $s \in S$ the number of items of size s is $n_s \in \mathbb{N}$. Let $n = \sum_{s \in S} n_s$ be the total number of items to be packed.

The First Fit algorithm implies the following: given any order of the items, assume that the kth bin is the first that has empty space of size at least s for some item size $s \in S$ in the end of the algorithm. Then all the items of size at most s are inserted in the first k bins.

Claim 1. *If $S = \{s_1, s_2, \ldots, s_t\}$ are the possible item sizes and s_j divides s_{j+1} for every $j = 1, 2, \ldots, t-1$, furthermore s_t divides the bin-size b, then the First Fit algorithm uses the same number of bins (namely $\lceil \frac{\sum_{i=1}^{t} n_{s_i} s_i}{b} \rceil$) for any ordering of the items.*

Proof. We prove a stronger claim by induction on t: for any ordering, the sum of the empty spaces of the bins is at most $b - s_1$. This implies the claim, since it means that the number of bins is at most $\frac{(b-s_1)+\sum_{i=1}^{t} n_{s_i} s_i}{b}$ which is strictly less than $\frac{\sum_{i=1}^{t} n_{s_i} s_i}{b} + 1$.

For $t = 1$ the claim is obviously true since there is at most one empty space, and its size is divisible by s_1. For general t, suppose that the k-th bin is the first that has empty space, it has size s, and i is the largest index such that $s_i \leq s$. It follows that the items of size at most s_i are all in the first k bins. We are done if $i = t$, otherwise consider all items that are not packed into the first k bins. These items have sizes $s_{i+1}, \ldots, s_t$, so by induction the sum of the empty spaces of the bins after the k-th is at most $b - s_{i+1}$. Thus the total sum of empty spaces is at most $s + b - s_{i+1}$ which is at most $b - s_1$ since $s_{i+1} - s \geq s_1$. □

We will concentrate on two special kinds of first fit algorithms. The algorithm for the monotone increasing order is called the *First Fit Increasing (FFI)* algorithm. It is easy to observe that after the FFI algorithm starts a new bin, it never puts items in earlier bins. This makes it very fast, but very inefficient. However, since we are interested in orderings that give the *worst* packing, we will use variants of the FFI algorithm. The efficiency of FFI packing compared to the worst possible first fit packing has been studied by Boyar et al. [7]. They showed that the number of bins in the worst first fit packing is at most $\frac{6}{5}$ times the number of bins in the FFI packing. If the item sizes are small compared to the bin size, a stronger bound can be given.

Theorem 2 ([7]). *For arbitrary item sizes, the number of bins in any first fit packing is at most $\frac{6}{5}$ times the number of bins given by the FFI algorithm plus 11. If every item has size at most $\frac{b}{r}$ for some integer r, then the ratio is asymptotically at most $\frac{r^2+r}{r^2+1}$.*

It follows from this result that by there is an easily computable upper bound on the number of bins needed ($\frac{6}{5} \times FFI + 11$) that overestimates the worst case by a multiplicative factor of at most $\frac{6}{5}$. Recently, Lin et al. [13] introduced a Modified First Fit Increasing Algorithm that gave an improved multiplicative factor of $\frac{80}{71}$. We will show in Sections 2.3 and 2.4 that for the item and bin sizes given in Table 1, there are easily computable upper bounds that overestimate the number of bins needed by at most 1.

The first fit algorithm for the monotone decreasing order, called the *First Fit Decreasing (FFD)* algorithm, performs much better, so we will use it for non-GMPLS demands.

If we have already packed the non-GMPLS demands then some bins have empty spaces that can be filled later when the GMPLS demands are being packed. Therefore we have a problem where the bin size is not the same number b for the bins, but it differs from one bin to the next. This motivates the following claims.

One of the difficulties of analyzing the first fit algorithm for arbitrary orderings is the lack of monotonicity: Graham [10] gave examples where increasing the size of one bin or deleting an item can actually *increase* the number of bins needed; see also [12], pages 66-67. However, this cannot happen for the FFI algorithm, as the following two claims show (the proofs are simply by induction, the details are left to the reader.)

Claim 3. *We are given a list of item sizes $z_1, \ldots, z_n$ in increasing order (the same size can appear multiple times), and we have different bin sizes: $b_1, \ldots, b_m$ (in the order of bins). We assume that the fist fit packing in that order can pack all items. Let $1 \leq k \leq n$. The first fit packing of $z_k, z_{k+1}, \ldots, z_n$ uses at most as many bins as the first fit packing of $z_1, \ldots, z_n$.* □

Claim 4. *We have different integer bin sizes: $b_1, \ldots, b_m$ (in the order of bins), and a list $z_1, \ldots, z_n$ of item sizes to pack (in increasing order). We assume that the FFI algorithm can pack all items into the bins; let k be the number of bins used. If we increase the size of a single bin by one (i.e. $b'_i = b_i + 1$ for some i), then for the new bin sizes the FFI algorithm uses at most k bins, too.* □

2.2 ODU1 and ODU2

The cases of ODU1 and ODU2 are simple by Claim 1. Since the item sizes are multiples of each other and the bin size is a multiple of the largest item size, every order gives the same number of bins: $\lceil \frac{\sum_{i=1}^{t} n_{s_i} s_i}{b} \rceil$.

2.3 ODU3

Here $b = 32$ and $S = \{1, 2, 8, 9, 32\}$. This case is special because of the presence of items of size 9. Items of size 32s always occupy a full bin, thus we can omit them.

Consider the following ordering of the items: first come those of size 1, then those of size 2, finally two items of size 8 followed by one item of size 9 iterated as long as this can be done when the rest of the items arrive (in any order: these are either all the same size 8 or 9, or there is one 8 and some 9s). This is not necessarily the worst ordering but we claim the following upper bound for the number of bins.

Claim 5. *Assume that in the ordering above, first we use the First Fit algorithm until we run out of items of size 1 and 2, and when we run out of these we close the bins used so far, and put the remaining items (using the above ordering) into new bins with the First Fit algorithm. This way we get an upper bound on the number of bins which is almost sharp: it is at most one more than the best upper bound.*

Proof. It is easy to see that this bound is at most one more than the number of bins needed when the First Fit algorithm runs for the same ordering. Indeed, the difference between the two algorithms is that First Fit may put some (at most 3) items of size 8 or 9 into the last bin that has items of size 1 or 2. If it puts 3 of them there, then the difference is one bin. If it puts less than 3, the difference can only be less.

The non-trivial part is to prove that the algorithm gives an upper bound for the First Fit algorithm running on any ordering. First suppose that there are no items of size 1 or 2. In this case any bin except for the last one will have 3 or 4 items, and in the latter case all four are 8s. It follows that our ordering is worst possible, because it has the least possible number of bins with four 8s.

Now suppose that there are items of size 1 or 2. Consider a First Fit packing for an arbitrary ordering. Let q be the empty space in the last bin. Let B_1 be the set of bins that only have items of size 8 or 9 in them; of the other bins, let B_2 denote the set of those that contain only items of size 1,2 and 8, and let B_3 be the rest. Let furthermore q_1, q_2, q_3 denote the sum of empty spaces in the bins in B_1, B_2, B_3 respectively, and let B_1', B_2' and q_1', q_2' denote the corresponding bins and empty spaces in the packing given by the upper bound algorithm ($B_3' = \emptyset$). To prove the claim, we have to prove that $q_1 + q_2 + q_3 \leq q_1' + q_2' + 31$. We distinguish three cases.

Case 1. The last bin is in B_1, and it contains at least one item of size 8. In this case we can use an argument similar to the one in the proof of Claim 1 to show that $q_2 + q_3 \leq 7 + |B_3|$. Indeed, $q_2 \leq 7$ if $q_3 = 0$, and the presence of a 9 in a bin can increase the empty space by at most one in that bin. Thus it is enough to prove that $q_1 + |B_3| \leq q_1' + q_2' + 24$. Since q is at most 24, it suffices to prove $q_1 + |B_3| - q \leq q_1' + q_2'$. Actually it is easy to see that $q_1 + |B_3| - q \leq q_1'$, if we observe that B_1' is constructed in a way that maximizes the empty space per item of size 9.

Case 2. The last bin is in B_1, and it does not contain items of size 8. Here q is at most 23, so we can use the above argument if $q_2 + q_3 \leq 8 + |B_3|$. The only situation when this is not true is the following: there is a bin in B_2 with empty space 1, and another with empty space 8. In this case $q_2 + q_3 = 9 + |B_3|$, so we have to prove $q_1 + |B_3| - q \leq q_1' + q_2' - 1$. Since an item of size 9 generates an empty space of size strictly more than 1 in B_1', the inequality holds if $B_3 \neq \emptyset$, and it also holds if $q_2' > 0$. The only remaining case is when $B_3 = 0$ and $q_2' = 0$, but the latter means that the number of items of size 1 is even, so there cannot be a bin in B_2 with empty space 1.

Case 3. The last bin is in $B_2 \cup B_3$. In this case the last bin contains a 1 or a 2. If it contains a 1, the total empty space is q which is strictly less than 32, so we are done. Otherwise let k be the last bin that contains a 1. Bin k has empty space at most one, the last bin contains empty space at most 30, and all other bins which contain empty space (of size at most one) also contain a 9. Since the 9s generate more than one empty space in B_1', this implies that $q_1 + q_2 + q_3 \leq q_1' + q_2' + 31$. □

2.4 ODU4

Here $b = 80$, and $S = \{1, 2, 8, 31, 80\}$. Again, we can assume here that there are no 80s. Similarly to the ODU3 case, consider the following ordering and algorithm: first come the items of size one, then the items of size 2, then the items of size 8, and all these are packed into bins using the First Fit algorithm (using $\lceil \frac{n_1+2n_2+8n_8}{80} \rceil$ bins by Claim 1). Then we close the bins used so far. Finally, the 31s are put into new bins, two items in each bin. This will use $\lceil \frac{n_1+2n_2+8n_8}{80} \rceil + \lceil \frac{n_{31}}{2} \rceil$ number of bins altogether.

Claim 6. *This is an upper bound, and it is at most one more than the best upper bound.*

Proof. As in Claim 5, it is easy to see that this gives at most one more than the First Fit algorithm for the same ordering. Let us show that it is an upper bound for any ordering.

Consider a First Fit packing for an arbitrary ordering. Let q be the empty space in the last bin. Let B_1 be the set of bins that only have items of size 31 in them; of the other bins, let B_2 denote the set of those that contain only items

of size 1,2 and 8, and let B_3 be the rest. Let furthermore q_1, q_2, q_3 denote the sum of empty spaces in the bins in B_1, B_2, B_3 respectively, and let B'_1, B'_2 and q'_1, q'_2 denote the corresponding bins and empty spaces in the packing given by the upper bound algorithm ($B'_3 = \emptyset$). To prove the claim, we have to prove that $q_1+q_2+q_3 \leq q'_1+q'_2+79$. We distinguish two cases.

Case 1. The last bin is in B_1. In this case we can use an argument similar to the one in the proof of Claim 1 to show that $q_2 + q_3 \leq 31 + 2|B_3|$. Indeed, $q_2 \leq 31$ if $q_3 = 0$, and the presence of a 31 in a bin can increase the empty space by at most two. Thus it is enough to prove that $q_1 + 2|B_3| \leq q'_1 + q'_2 + 48$. Since q is at most 49, it suffices to prove $q_1 + 2|B_3| - q \leq q'_1 + q'_2 - 1$. Actually it is easy to see that $q_1 + 2|B_3| - q \leq q'_1$, if we observe that B'_1 is constructed in a way that maximizes the empty space per item of size 31. Equality cannot hold if $B_3 \neq \emptyset$, because items of size 31 generate an empty space of size more than 2 in B'_1. On the other hand, equality is also impossible if $B_3 = \emptyset$, because in that case $q_1 = q'_2$, and obviously $q > 0$ (since the last bin is in B_1). Hence we proved that $q_1 + 2|B_3| - q \leq q'_1 - 1$.

Case 2. The last bin is in $B_2 \cup B_3$. In this case $B_1 = \emptyset$, and the last bin contains a 1, 2 or 8. If it contains a 1, the total empty space is q which is strictly less than 80, so we are done. If it contains a 2, then $q \leq 78$. Furthermore, the earlier bins contain empty space at most 1, and at most $1 + |B_3|$ of them contain 1. Since $q'_1 \geq |B_3|$, this implies that $q_1 + q_2 + q_3 \leq 79 + |B_3| \leq q'_1 + q'_2 + 79$.

If the last bin contains only 8s and possibly 31s, then $q \leq 72$. Furthermore, the total sum of empty space in earlier bins is at most $7 + 2|B_3|$. Since $q'_1 \geq 2|B_3|$, this implies that $q_1 + q_2 + q_3 \leq q'_1 + q'_2 + 79$. □

3 Inclusion of non-GMPLS demands

In the case of non-GMPLS demands, the items are not packed automatically into bins using the First Fit algorithm. Instead, they can be packed arbitrarily into the component links before the first-fit packing of GMPLS demands.

A natural approach is to try to pack the non-GMPLS demands into as few bins as possible. This corresponds to the classical bin-packing problem with a restricted class of item sizes. By Claim 1, in case of ODU1 and ODU2 the first fit algorithm (with any order) gives the best possible packing. For ODU3 and ODU4, the first fit algorithm with decreasing order (First Fit Decreasing) gives results very close to optimal. Even for ODUFlex, it gives at most $\frac{11}{9}OPT + \frac{6}{9}$ [8]. The following example shows that FFD might not be optimal for ODU3: the optimal packing is 48 bins containing items $9, 9, 8, 2, 2, 2$, but FFD for this input will use 49 bins altogether.

In addition, the following claim shows that the increasing order always gives the largest number of bins if the item sizes divide each other.

Claim 7. *Let $S = \{s_1, s_2, \ldots, s_t\}$ be the item sizes, and let us assume that s_j divides s_{j+1} for every $j = 1, 2, \ldots, t-1$. We have different integer bin sizes: $b_1, \ldots, b_m$ (in the order of bins). The the increasing order uses the most bins amongst all the possible ordering of items.*

Proof. The proof is by induction on $\sum n_{s_i} s_i^2$, the details are omitted. □

Note that if the number of different item sizes is fixed (t), then there is a polynomial algorithm by Jansen and Solis-Oba [11] that computes a packing that uses at most 1 more bins than the optimal packing. However, the running time of this algorithm is $O(t^{21t} 2^{2^{3t+3}} (\log^2 n + \log b)^4)$, which is unusable in practice. In contrast, the FFD algorithm gives close to optimal results in almost linear time.

Given a packing of the non-GMPLS demands, the problem of estimating the number of bins needed for the GMPLS demands corresponds to maximum resource bin packing with variable bin sizes. The difficulty of this problem depends on the ODU type of the link.

- For ODU1 and ODU2, we can pack both the non-GMPLS and GMPLS items using first fit with arbitrary ordering, since the number of bins used will always be $\lceil \frac{\sum_{i=1}^{t} n_{s_i} s_i}{b} \rceil$ by Claim 1. This means that the sharp bound can be very easily computed.

- For ODU3, we can use FFI if there are no ODU2e GMPLS items. In this case the item sizes are multiples of each other, so by Claim 7 we obtain the worst possible first fit packing of GMPLS items.

  If there are ODU2e GMPLS items, FFI does not necessarily give the worst packing. In this case we use the algorithm in Section 2.3 to estimate the worst packing of GMPLS items (i.e. 1s and 2s first using the possible partial bins, then close the partially filled bins and continue that algorithm). We note that "closing

the bins" after packing the items of size 1 and 2 is increasing the number of bins by at most 1, since only the last closed bin may contain an empty space of size at least 8.

- For ODU4, we can use the algorithm in Section 2.4 to estimate the worst packing of GMPLS items. As in the ODU3 case, "closing the bins" after packing the items of size 1, 2 and 8 is increasing the number of bins by at most 1, since only the last closed bin may contain an empty space of size at least 31.

4 Concluding Remarks

We conclude our work with mentioning two possible further extensions of the model above.

Firstly, an extension could not have been discussed due to lack of space is the multiple phase version of the problem. Assume we want to dimension a fault tolerant network, i.e. certain consecutive failure scenarios must also be taken into account. As a result of a failure, some already allocated items may be dropped, but some others arrive (failed LSPs are released and backup LSPs are set up). Again, we know which items are removed and what kind of new items arrive, but we do not know the order of the insertions and removals. In case of multiple failure analysis, this second phase may be followed by some limited number of additional ones. Using the techniques presented in the paper, similar worst case estimations can be obtained for this case, as well.

Secondly, in this paper, we have investigated a particular (the worst possible) ordering of the items and gave a (sharp) upper bound. In practice, however, it is very unlikely that the items arrive in this particularly bad ordering. Instead, it would make sense to give an estimation on the number of used bins that is an upper bound with high probability. This is an interesting subject of further work.

References

[1] ITU-T, *Interfaces for the Optical Transport Network (OTN)*, G.709 Recommendation, December 2009.

[2] ITU-T, *Architecture of optical transport networks*, G.872 Recommendation, November 2001.

[3] ITU-T, *Transport of IEEE 10G Base-R in Optical Transport Networks*, Supplement G.Sup43, June 2010.

[4] D. Papadimitriou, Ed., *Generalized Multi-Protocol Label Switching (GMPLS) Signaling Extensions for G.709 Optical Transport Networks Control*, RFC 4328, Jan 2006.

[5] Mannie, E., *Generalized Multi-Protocol Label Switching (GMPLS) Architecture*, RFC 3945, October 2004.

[6] T. Kárász, L. Bajzik, Z. Lakatos, T. Jakab, *Packing Problems in Dimensioning of GMPLS-Controlled SDH Networks*, in Proceedings of NETWORKS 2010, Warsaw, Poland, September 2010.

[7] J. Boyar, L. Epstein, L.M. Favrholdt, J.S. Kohrt, K.S. Larsen, M.M. Pedersen, S. Wøhlk, *The maximum resource bin packing problem*, Theoretical Computer Science 362 (2006), 127–139.

[8] G. Dósa, *The tight bound of first fit decreasing bin-packing algorithm is* $FFD(L) \leq \frac{11}{9}OPT(L) + \frac{6}{9}$, Lecture Notes in Computer Science 4614 (2007), 1–11.

[9] L. Gai, G. Zhang, *Hardness of lazy packing and covering*, Operations Research Letters 37:2 (2009), 89–92.

[10] R.L. Graham, *Bounds on multiprocessing anomalies and related packing algorithms*, AFIPS Joint Computer Conferences, Proceedings of the May 16-18, 1972, spring joint computer conference, 205–217.

[11] K. Jansen, R. Solis-Oba, *An OPT+1 Algorithm for the Cutting Stock Problem with Constant Number of Object Lengths*, Proceedings of IPCO 2010, LNCS 6080 (2010), 438–449.

[12] D.S. Johnson, *Near-optimal bin packing algorithms*, Ph.D. Dissertation, Massachusetts Institute of Technology, 1973.

[13] M. Lin, Y. Yang, J. Xu, *Improved Approximation Algorithms for Maximum Resource Bin Packing and Lazy Bin Covering Problems*, Algorithmica 57:2 (2010), 232–251.

[14] B. Xia, Z. Tan, *Tighter bounds of the First Fit algorithm for the bin-packing problem*, Discrete Applied Mathematics 158 (2010), 1668–1675.